# Lumen AI Agentic Companion: Technical Framework

## 1. System Architecture

### 1.1 Core Components

#### 1.1.1 Lumen Core Engine

* **Purpose**: Central processing unit for Lumen's intelligence and behavior
* **Responsibilities**:
  + Orchestrate interactions between all Lumen subsystems
  + Maintain context awareness across app features
  + Process user signals and determine appropriate responses
  + Manage Lumen's appearance and behavior states
* **Integration Points**:
  + Interfaces with all app features via the Feature Integration Layer
  + Connects to User Context Service for personalization
  + Communicates with AI Service for intelligent responses

#### 1.1.2 User Context Service

* **Purpose**: Build and maintain comprehensive user context model
* **Responsibilities**:
  + Track user's philosophical interests and exploration history
  + Monitor engagement patterns and preferences
  + Build and update user's philosophical profile
  + Detect context changes that should trigger Lumen
* **Data Model**:
  + PhilosophicalProfile: User's interests, strengths, and exploration patterns
  + EngagementHistory: Record of feature usage and content interaction
  + PreferenceSettings: User's explicit settings for Lumen behavior
  + ContextualState: Current app state and user activity

#### 1.1.3 Intervention Manager

* **Purpose**: Determine when and how Lumen should appear
* **Responsibilities**:
  + Apply intervention rules based on current context
  + Balance helpfulness against intrusiveness
  + Schedule and prioritize potential interventions
  + Adapt intervention frequency based on user feedback
* **Key Algorithms**:
  + Intervention Opportunity Detection
  + Intervention Value Assessment
  + Timing Optimization
  + Frequency Regulation

#### 1.1.4 Content Generation Engine

* **Purpose**: Create contextually relevant philosophical content
* **Responsibilities**:
  + Generate philosophical questions and prompts
  + Create guidance suggestions for navigation
  + Produce reflection prompts and insights
  + Craft social and sharing suggestions
* **Integration Points**:
  + Connects to AI Service for content generation
  + Interfaces with Concept Database for philosophical knowledge
  + Links to User Context Service for personalization

#### 1.1.5 Visual Presentation Manager

* **Purpose**: Control Lumen's visual appearance and animations
* **Responsibilities**:
  + Manage Lumen's visual state and evolution
  + Control animations and transitions
  + Position Lumen appropriately on screen
  + Adapt visual presentation based on context
* **Visual States**:
  + Dormant: Minimized or hidden state
  + Ambient: Subtle presence indicating availability
  + Active: Fully engaged when providing guidance
  + Celebratory: Enhanced appearance for achievements
  + Evolving: Visual changes reflecting user progression

### 1.2 Supporting Services

#### 1.2.1 AI Service

* **Purpose**: Provide AI capabilities to all Lumen components
* **Responsibilities**:
  + Process natural language for understanding user intent
  + Generate contextually appropriate responses
  + Analyze philosophical content and relationships
  + Learn from user interactions to improve personalization
* **AI Capabilities**:
  + Natural Language Understanding
  + Context-Aware Response Generation
  + Philosophical Concept Mapping
  + User Preference Learning

#### 1.2.2 Concept Database Service

* **Purpose**: Provide access to philosophical knowledge
* **Responsibilities**:
  + Maintain structured database of philosophical concepts
  + Track relationships between concepts
  + Map user exploration to concept space
  + Provide relevant concept information for guidance
* **Data Model**:
  + Concept: Individual philosophical concepts
  + ConceptRelationship: Connections between concepts
  + ConceptHierarchy: Organizational structure of concepts
  + UserConceptInteraction: History of user engagement with concepts

#### 1.2.3 Notification Service

* **Purpose**: Manage external communications from Lumen
* **Responsibilities**:
  + Schedule and deliver push notifications
  + Generate notification content
  + Track notification engagement
  + Optimize notification timing and frequency
* **Notification Types**:
  + Philosophical Prompts: Thought-provoking questions
  + Discovery Alerts: New content or feature announcements
  + Re-engagement Invitations: Personalized return prompts
  + Achievement Celebrations: Recognition of milestones

#### 1.2.4 Analytics Service

* **Purpose**: Track and analyze Lumen's effectiveness
* **Responsibilities**:
  + Collect interaction metrics
  + Analyze engagement impact
  + Identify successful intervention patterns
  + Support A/B testing of Lumen behaviors
* **Key Metrics**:
  + Intervention Acceptance Rate
  + Feature Discovery Attribution
  + Engagement Lift Measurement
  + User Satisfaction Indicators

### 1.3 Integration Layer

#### 1.3.1 Feature Integration Layer

* **Purpose**: Enable Lumen to interact with all app features
* **Responsibilities**:
  + Provide standardized interfaces to all features
  + Maintain context when transitioning between features
  + Enable feature-specific Lumen behaviors
  + Support cross-feature guidance
* **Integration Interfaces**:
  + ILumenAskFeatureIntegration
  + ILumenExploreFeatureIntegration
  + ILumenQuestFeatureIntegration
  + ILumenJournalFeatureIntegration
  + ILumenForumFeatureIntegration

#### 1.3.2 UI Integration Layer

* **Purpose**: Manage Lumen's presence in the app UI
* **Responsibilities**:
  + Handle rendering of Lumen across app screens
  + Manage animations and transitions
  + Ensure consistent visual appearance
  + Support accessibility requirements
* **UI Components**:
  + LumenOrbComponent: Main visual representation
  + LumenDialogueComponent: Text and interaction interface
  + LumenAnimationController: Animation management
  + LumenAccessibilityProvider: Accessibility support

#### 1.3.3 Event Bus

* **Purpose**: Facilitate communication between Lumen components
* **Responsibilities**:
  + Broadcast system events
  + Enable loose coupling between components
  + Support asynchronous communication
  + Maintain event history for context
* **Event Types**:
  + User Interaction Events
  + Context Change Events
  + Intervention Events
  + System State Events

## 2. Functional Modules

### 2.1 Navigation & Guidance Module

#### 2.1.1 Path Recommendation Engine

* **Purpose**: Suggest optimal exploration paths
* **Functionality**:
  + Analyze concept map for relevant paths
  + Consider user history and interests
  + Identify unexplored areas of potential interest
  + Generate personalized exploration recommendations
* **Integration Points**:
  + Explore Feature: Highlight paths on concept map
  + User Context Service: Access user interests
  + Concept Database: Map philosophical relationships

#### 2.1.2 Feature Discovery Guide

* **Purpose**: Introduce users to relevant features
* **Functionality**:
  + Identify underutilized features matching user interests
  + Create contextual introduction to new features
  + Guide users through feature first-use
  + Track feature adoption and usage
* **Integration Points**:
  + All app features via Feature Integration Layer
  + User Context Service: Track feature familiarity
  + Analytics Service: Measure feature adoption

#### 2.1.3 Learning Path Optimizer

* **Purpose**: Create optimal philosophical learning sequences
* **Functionality**:
  + Analyze concept prerequisites and relationships
  + Map user's current knowledge state
  + Identify knowledge gaps and learning opportunities
  + Generate personalized learning pathways
* **Integration Points**:
  + Concept Database: Access concept relationships
  + User Context Service: Track learning progress
  + Quest Feature: Align with quest progression

### 2.2 Contextual Engagement Module

#### 2.2.1 Philosophical Prompt Generator

* **Purpose**: Create thought-provoking philosophical prompts
* **Functionality**:
  + Generate contextually relevant philosophical questions
  + Adapt prompt complexity to user's philosophical sophistication
  + Create prompts that connect to current content
  + Vary prompt types to maintain engagement
* **Integration Points**:
  + AI Service: Generate philosophical content
  + User Context Service: Personalize complexity
  + Content Generation Engine: Format and deliver prompts

#### 2.2.2 Content Enhancement Provider

* **Purpose**: Enrich user's engagement with philosophical content
* **Functionality**:
  + Provide additional context for concepts
  + Suggest related concepts and connections
  + Offer multiple perspectives on philosophical topics
  + Generate examples and applications
* **Integration Points**:
  + Concept Database: Access philosophical knowledge
  + Ask Feature: Enhance AI responses
  + Explore Feature: Enrich concept exploration

#### 2.2.3 Reflection Trigger System

* **Purpose**: Prompt meaningful philosophical reflection
* **Functionality**:
  + Identify opportunities for reflection
  + Generate personalized reflection prompts
  + Connect current content to past explorations
  + Apply Socratic questioning techniques
* **Integration Points**:
  + Journal Feature: Suggest reflection topics
  + User Context Service: Access exploration history
  + AI Service: Generate reflection questions

### 2.3 Social Connection Module

#### 2.3.1 Community Integration Engine

* **Purpose**: Connect users to relevant community activity
* **Functionality**:
  + Identify forum discussions matching user interests
  + Suggest sharing opportunities at meaningful moments
  + Highlight community engagement opportunities
  + Create contextual bridges to social features
* **Integration Points**:
  + Forum Feature: Access discussion data
  + User Context Service: Match interests to discussions
  + Content Generation Engine: Create sharing prompts

#### 2.3.2 Collaborative Learning Facilitator

* **Purpose**: Enable learning from and with other users
* **Functionality**:
  + Identify peer learning opportunities
  + Suggest discussion topics based on exploration
  + Facilitate philosophical dialogues
  + Create group exploration recommendations
* **Integration Points**:
  + Forum Feature: Enable group discussions
  + User Context Service: Match users with similar interests
  + Concept Database: Identify discussion-worthy concepts

#### 2.3.3 Achievement Sharing System

* **Purpose**: Facilitate meaningful sharing of philosophical progress
* **Functionality**:
  + Identify shareable achievements and insights
  + Generate shareable content from user's journey
  + Create contextual sharing prompts
  + Track sharing impact and engagement
* **Integration Points**:
  + Forum Feature: Enable sharing functionality
  + User Context Service: Identify meaningful achievements
  + Content Generation Engine: Create shareable content

### 2.4 Notification & Nudge Module

#### 2.4.1 Intelligent Notification Manager

* **Purpose**: Deliver valuable philosophical notifications
* **Functionality**:
  + Generate personalized notification content
  + Optimize notification timing
  + Track notification effectiveness
  + Balance engagement with respect for user attention
* **Integration Points**:
  + Notification Service: Deliver notifications
  + User Context Service: Personalize content
  + Analytics Service: Track effectiveness

#### 2.4.2 Contextual Nudge System

* **Purpose**: Provide subtle in-app guidance
* **Functionality**:
  + Identify opportunities for valuable nudges
  + Generate non-intrusive nudge content
  + Position nudges appropriately in UI
  + Track nudge effectiveness
* **Integration Points**:
  + UI Integration Layer: Display nudges
  + User Context Service: Personalize nudges
  + Analytics Service: Measure nudge impact

#### 2.4.3 Re-engagement Campaign Manager

* **Purpose**: Bring users back to their philosophical journey
* **Functionality**:
  + Identify inactive users
  + Generate personalized re-engagement content
  + Create "philosophical question of the day"
  + Schedule re-engagement notifications
* **Integration Points**:
  + Notification Service: Deliver re-engagement messages
  + User Context Service: Personalize based on history
  + Analytics Service: Track re-engagement effectiveness

### 2.5 Onboarding & User Flow Module

#### 2.5.1 Adaptive Onboarding Guide

* **Purpose**: Personalize the onboarding experience
* **Functionality**:
  + Tailor onboarding to philosophical interests
  + Progressively introduce features
  + Provide contextual help for first-time usage
  + Track onboarding completion and effectiveness
* **Integration Points**:
  + All features via Feature Integration Layer
  + User Context Service: Store onboarding progress
  + Analytics Service: Measure onboarding effectiveness

#### 2.5.2 Flow Optimization Engine

* **Purpose**: Improve user journeys through the app
* **Functionality**:
  + Identify friction points in user flows
  + Provide assistance at challenging moments
  + Guide users through complex processes
  + Suggest optimal next steps
* **Integration Points**:
  + All features via Feature Integration Layer
  + User Context Service: Track user journey
  + Analytics Service: Identify friction points

#### 2.5.3 Feature Transition Guide

* **Purpose**: Enable smooth transitions between features
* **Functionality**:
  + Maintain context when switching features
  + Suggest cross-feature actions
  + Create coherent philosophical narratives across features
  + Guide users through feature transitions
* **Integration Points**:
  + All features via Feature Integration Layer
  + User Context Service: Maintain cross-feature context
  + Content Generation Engine: Create transition prompts

### 2.6 Gamification & Progression Module

#### 2.6.1 Achievement Recognition System

* **Purpose**: Acknowledge philosophical milestones
* **Functionality**:
  + Identify meaningful achievements
  + Award titles and badges
  + Provide contextual celebration
  + Track progression milestones
* **Integration Points**:
  + User Context Service: Track achievements
  + Visual Presentation Manager: Display celebrations
  + Content Generation Engine: Create achievement messages

#### 2.6.2 Visual Evolution Controller

* **Purpose**: Evolve Lumen's appearance based on user progression
* **Functionality**:
  + Define visual evolution stages
  + Map user progress to visual changes
  + Implement gradual visual transformation
  + Create meaningful visual feedback
* **Integration Points**:
  + Visual Presentation Manager: Implement visual changes
  + User Context Service: Track progression
  + UI Integration Layer: Render evolved appearance

#### 2.6.3 Motivation Enhancement Engine

* **Purpose**: Maintain engagement through intrinsic motivation
* **Functionality**:
  + Identify user's philosophical motivations
  + Balance challenge and achievement
  + Create meaningful goals
  + Provide motivational feedback
* **Integration Points**:
  + User Context Service: Understand motivations
  + Content Generation Engine: Create motivational content
  + Analytics Service: Track motivation effectiveness

## 3. Data Models

### 3.1 User Context Models

#### 3.1.1 PhilosophicalProfile

{

userId: string,

interests: {

area: string,

strength: number,

lastEngaged: timestamp

}[],

knowledgeAreas: {

concept: string,

familiarity: number,

lastEngaged: timestamp

}[],

preferredLearningStyle: string,

sophisticationLevel: number,

motivationFactors: string[]

}

#### 3.1.2 EngagementHistory

{

userId: string,

featureEngagement: {

featureId: string,

engagementCount: number,

lastEngaged: timestamp,

averageSessionDuration: number

}[],

conceptEngagement: {

conceptId: string,

engagementCount: number,

lastEngaged: timestamp,

favorited: boolean

}[],

questionHistory: {

questionId: string,

question: string,

timestamp: timestamp,

relatedConcepts: string[]

}[]

}

#### 3.1.3 PreferenceSettings

{

userId: string,

lumenEnabled: boolean,

interventionFrequency: "low" | "medium" | "high",

notificationsEnabled: boolean,

preferredInterventionTypes: string[],

disabledFeatures: string[],

accessibilityPreferences: {

screenReader: boolean,

reducedMotion: boolean,

highContrast: boolean

}

}

#### 3.1.4 ContextualState

{

userId: string,

currentFeature: string,

currentScreen: string,

currentActivity: string,

activeConcepts: string[],

sessionDuration: number,

lastInteraction: timestamp,

currentFocus: string

}

### 3.2 Intervention Models

#### 3.2.1 InterventionOpportunity

{

opportunityId: string,

userId: string,

triggerType: string,

triggerContext: object,

estimatedValue: number,

suggestedTiming: timestamp,

expirationTime: timestamp,

interventionType: string,

priority: number

}

#### 3.2.2 InterventionRecord

{

interventionId: string,

userId: string,

opportunityId: string,

interventionType: string,

content: object,

timestamp: timestamp,

userResponse: "accepted" | "dismissed" | "ignored",

effectiveness: number,

followupActions: string[]

}

#### 3.2.3 InterventionRule

{

ruleId: string,

triggerConditions: {

contextualFactors: object,

userFactors: object,

temporalFactors: object

},

interventionType: string,

priority: number,

cooldownPeriod: number,

personalizable: boolean

}

### 3.3 Content Models

#### 3.3.1 PhilosophicalPrompt

{

promptId: string,

promptType: "question" | "insight" | "reflection" | "challenge",

content: string,

complexity: number,

relatedConcepts: string[],

appropriateContexts: string[],

followupPrompts: string[]

}

#### 3.3.2 GuidanceSuggestion

{

suggestionId: string,

suggestionType: "navigation" | "feature" | "social" | "reflection",

content: string,

targetFeature: string,

targetConcept: string,

relevanceFactors: object,

expirationConditions: object

}

#### 3.3.3 LumenDialogue

{

dialogueId: string,

dialogueType: "greeting" | "guidance" | "insight" | "celebration",

content: string,

tone: string,

animations: string[],

duration: number,

priority: number

}

### 3.4 Visual Models

#### 3.4.1 LumenVisualState

{

stateId: string,

baseAppearance: string,

glowIntensity: number,

particleEffects: string[],

animations: string[],

colorPalette: {

primary: string,

secondary: string,

accent: string

},

evolutionStage: number

}

#### 3.4.2 LumenAnimation

{

animationId: string,

animationType: "idle" | "appear" | "celebrate" | "guide" | "transition",

keyframes: object[],

duration: number,

easing: string,

particleEffects: object,

soundEffects: string[]

}

#### 3.4.3 LumenPositioning

{

positioningId: string,

contextType: string,

anchorElement: string,

offsetX: number,

offsetY: number,

zIndex: number,

entryDirection: string,

exitDirection: string

}

## 4. API Specifications

### 4.1 Lumen Core API

#### 4.1.1 Initialization and Configuration

// Initialize Lumen system

initializeLumen(config: LumenConfig): Promise<void>

// Update Lumen configuration

updateLumenConfig(config: Partial<LumenConfig>): Promise<void>

// Enable/disable Lumen

setLumenEnabled(enabled: boolean): Promise<void>

#### 4.1.2 Intervention Control

// Trigger Lumen to appear with specific guidance

triggerLumenIntervention(interventionType: string, context: object): Promise<InterventionResult>

// Dismiss current Lumen intervention

dismissCurrentIntervention(reason: string): Promise<void>

// Check if intervention is appropriate

canInterventionOccur(interventionType: string, context: object): Promise<boolean>

#### 4.1.3 User Interaction

// Handle user response to Lumen

handleLumenInteraction(interactionType: string, data: object): Promise<void>

// Record explicit user feedback

recordLumenFeedback(rating: number, feedback: string): Promise<void>

// Update user preferences for Lumen

updateLumenPreferences(preferences: Partial<LumenPreferences>): Promise<void>

### 4.2 Feature Integration API

#### 4.2.1 Ask Feature Integration

// Register Ask feature with Lumen

registerAskFeature(config: AskFeatureConfig): Promise<void>

// Notify of new question being asked

notifyNewQuestion(question: string, context: object): Promise<void>

// Get philosophical prompts for Ask feature

getAskPrompts(context: object): Promise<PhilosophicalPrompt[]>

// Enhance AI response with Lumen insights

enhanceAIResponse(response: string, context: object): Promise<EnhancedResponse>

#### 4.2.2 Explore Feature Integration

// Register Explore feature with Lumen

registerExploreFeature(config: ExploreFeatureConfig): Promise<void>

// Get path recommendations

getExplorationPaths(currentConcept: string, context: object): Promise<PathRecommendation[]>

// Highlight concepts on map

highlightRecommendedConcepts(context: object): Promise<ConceptHighlight[]>

// Get concept enhancement content

getConceptEnhancements(conceptId: string): Promise<ConceptEnhancement>

#### 4.2.3 Quest Feature Integration

// Register Quest feature with Lumen

registerQuestFeature(config: QuestFeatureConfig): Promise<void>

// Get quest guidance

getQuestGuidance(questId: string, progress: object): Promise<QuestGuidance>

// Celebrate quest milestone

celebrateQuestMilestone(milestoneType: string, context: object): Promise<CelebrationContent>

// Recommend next quest

getQuestRecommendations(context: object): Promise<QuestRecommendation[]>

#### 4.2.4 Journal Feature Integration

// Register Journal feature with Lumen

registerJournalFeature(config: JournalFeatureConfig): Promise<void>

// Get reflection prompts

getReflectionPrompts(context: object): Promise<ReflectionPrompt[]>

// Analyze journal entry

analyzeJournalEntry(entryText: string): Promise<JournalInsight>

// Connect journal to concepts

getRelatedConcepts(entryText: string): Promise<ConceptReference[]>

#### 4.2.5 Forum Feature Integration

// Register Forum feature with Lumen

registerForumFeature(config: ForumFeatureConfig): Promise<void>

// Get discussion recommendations

getDiscussionRecommendations(context: object): Promise<DiscussionRecommendation[]>

// Generate sharing content

generateSharingContent(contentType: string, context: object): Promise<SharingContent>

// Find philosophical connections

findPhilosophicalConnections(discussionId: string): Promise<PhilosophicalConnection[]>

### 4.3 Notification API

#### 4.3.1 Notification Management

// Schedule philosophical notification

schedulePhilosophicalNotification(type: string, context: object): Promise<NotificationId>

// Cancel scheduled notification

cancelNotification(notificationId: string): Promise<void>

// Track notification engagement

trackNotificationEngagement(notificationId: string, action: string): Promise<void>

#### 4.3.2 Nudge Management

// Register nudge opportunity

registerNudgeOpportunity(nudgeType: string, context: object): Promise<NudgeId>

// Display nudge

displayNudge(nudgeId: string): Promise<NudgeResult>

// Track nudge effectiveness

trackNudgeEffectiveness(nudgeId: string, result: string): Promise<void>

### 4.4 Analytics API

#### 4.4.1 Interaction Tracking

// Track Lumen interaction

trackLumenInteraction(interactionType: string, details: object): Promise<void>

// Track feature discovery

trackFeatureDiscovery(featureId: string, discoveryMethod: string): Promise<void>

// Track philosophical engagement

trackPhilosophicalEngagement(engagementType: string, details: object): Promise<void>

#### 4.4.2 Effectiveness Measurement

// Record intervention effectiveness

recordInterventionEffectiveness(interventionId: string, metrics: object): Promise<void>

// Get intervention analytics

getInterventionAnalytics(filters: object): Promise<InterventionAnalytics>

// Run A/B test for intervention

runInterventionABTest(testConfig: ABTestConfig): Promise<TestId>

## 5. Implementation Considerations

### 5.1 Technical Dependencies

#### 5.1.1 Required Services

* **AI Service**: Natural language processing and generation capabilities
* **User Profile System**: Access to user data and preferences
* **Notification System**: Push notification capabilities
* **Analytics Platform**: Tracking and measurement infrastructure
* **Content Management System**: Access to philosophical content

#### 5.1.2 Frontend Requirements

* **React Native Components**: For mobile UI rendering
* **Animation Library**: For Lumen's fluid animations
* **State Management**: For maintaining Lumen's state across the app
* **Accessibility Tools**: For ensuring inclusive experience

#### 5.1.3 Backend Requirements

* **API Framework**: For Lumen service endpoints
* **Database System**: For storing Lumen-related data
* **Task Scheduling**: For notification and intervention timing
* **Caching System**: For performance optimization

### 5.2 Performance Considerations

#### 5.2.1 Mobile Optimization

* Minimize battery impact through efficient animations
* Reduce network requests by caching common interactions
* Optimize rendering performance for smooth animations
* Implement lazy loading for non-critical components

#### 5.2.2 Responsiveness

* Ensure Lumen responds within 100ms to user interactions
* Implement optimistic UI updates for perceived performance
* Pre-compute common guidance suggestions for instant access
* Use background processing for complex operations

#### 5.2.3 Offline Capabilities

* Cache philosophical content for offline access
* Queue interactions for synchronization when online
* Maintain core functionality without network connection
* Gracefully degrade advanced features when offline

### 5.3 Scalability Considerations

#### 5.3.1 User Scale

* Design for efficient handling of millions of users
* Implement sharding for user context data
* Use distributed processing for AI operations
* Optimize database queries for high-volume access

#### 5.3.2 Content Scale

* Support growing philosophical concept database
* Implement efficient indexing for concept relationships
* Design for thousands of potential interventions
* Scale notification system for high-volume delivery

#### 5.3.3 Feature Scale

* Architecture supports addition of new app features
* Standardized integration interfaces for new features
* Extensible intervention rule system
* Modular design for adding new Lumen capabilities

### 5.4 Security and Privacy

#### 5.4.1 Data Protection

* Encrypt sensitive user philosophical profile data
* Implement proper access controls for user context
* Secure API endpoints with authentication
* Regular security audits for all Lumen components

#### 5.4.2 Privacy Controls

* Clear user controls for Lumen data collection
* Transparency about how user data informs Lumen
* Options to limit data usage for personalization
* Compliance with relevant privacy regulations

#### 5.4.3 Ethical Considerations

* Avoid manipulative engagement techniques
* Respect user attention and cognitive load
* Ensure philosophical guidance is balanced and diverse
* Prevent reinforcement of philosophical filter bubbles

## 6. Phased Implementation Plan

### 6.1 Phase 1: Foundation (Weeks 1-4)

* Implement Lumen Core Engine
* Develop basic User Context Service
* Create initial Visual Presentation Manager
* Integrate with Explore feature for basic navigation
* Implement fundamental intervention rules

### 6.2 Phase 2: Core Features (Weeks 5-8)

* Expand feature integrations to Ask and Journal
* Enhance Content Generation Engine
* Implement basic Notification integration
* Develop initial philosophical prompts
* Create first visual evolution stages

### 6.3 Phase 3: Advanced Capabilities (Weeks 9-12)

* Integrate with Quest and Forum features
* Implement advanced AI capabilities
* Develop comprehensive intervention rules
* Create sophisticated visual evolutions
* Enhance personalization capabilities

### 6.4 Phase 4: Refinement (Weeks 13-16)

* Implement A/B testing framework
* Optimize performance across all components
* Enhance accessibility features
* Refine intervention timing and frequency
* Develop advanced analytics dashboards

## 7. Testing Strategy

### 7.1 Unit Testing

* Test individual Lumen components in isolation
* Verify correct behavior of intervention rules
* Validate content generation algorithms
* Ensure proper state management

### 7.2 Integration Testing

* Test Lumen integration with each app feature
* Verify cross-feature functionality
* Validate notification system integration
* Test analytics data flow

### 7.3 User Experience Testing

* Conduct usability testing for Lumen interactions
* Measure perceived helpfulness and intrusiveness
* Test accessibility with diverse users
* Validate visual design and animations

### 7.4 Performance Testing

* Measure impact on app performance
* Test battery consumption
* Validate offline capabilities
* Stress test with high intervention volume